Adaptivity and Resource Control in Embedded Systems

Karl-Erik Årzén

Lund University

ACTORS:
Adaptivity & Control of Resources in Embedded Systems

LUND UNIVERSITY

Adaptivity and Resource Control in Cyber-Physical Systems

Karl-Erik Å

Lund

US Version

ACTORS:
Adaptivity & Control of
Resources in Embedded Systems

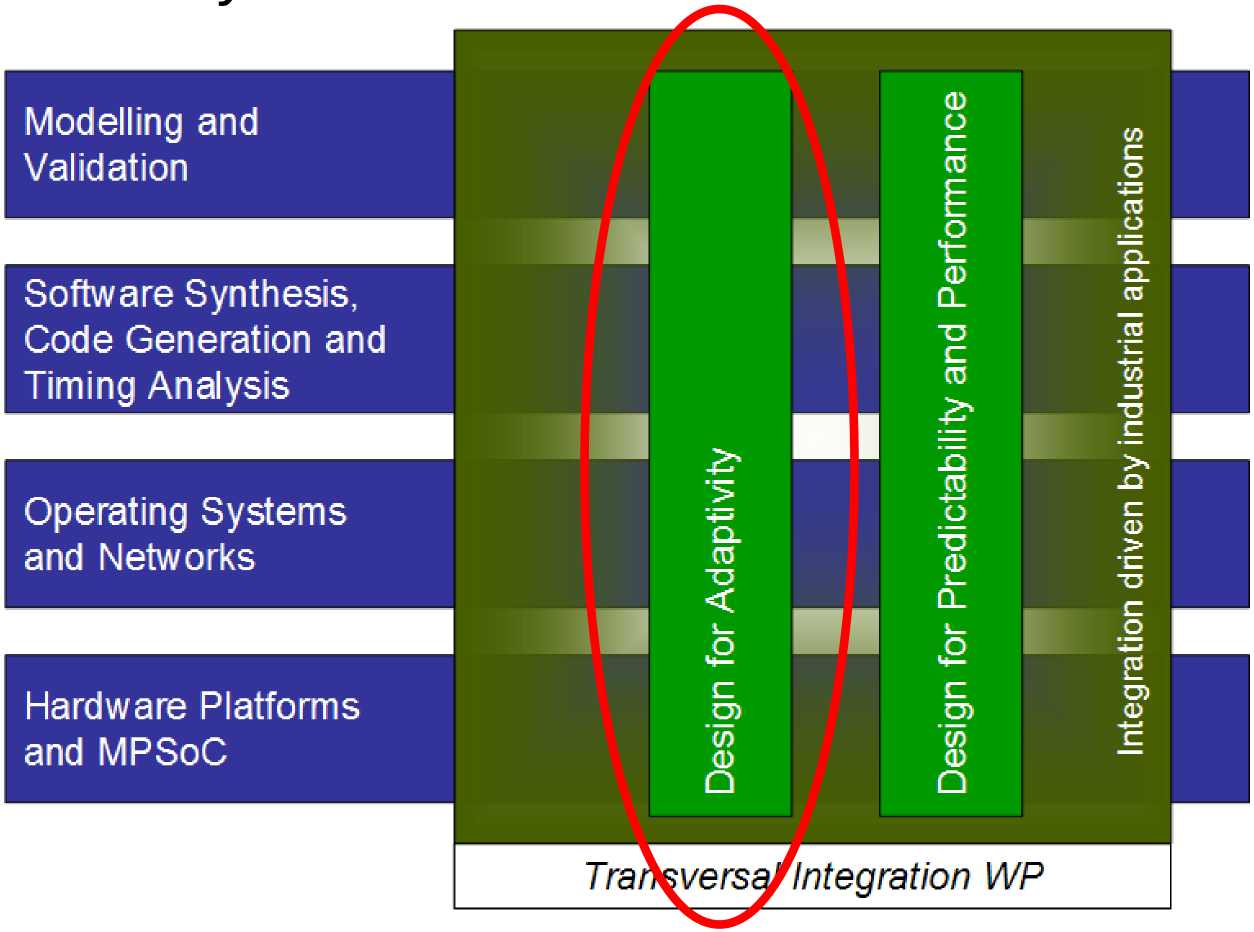
LUND UNIVERSITY

Outline

- **Adaptivity in Embedded Systems**
- ACTORS - Resource Management for Multimedia Dataflow Applications on Multi-core Platforms

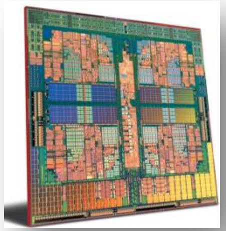
Artist

- ArtistDesign - European Network of Excellence on Design of Embedded System

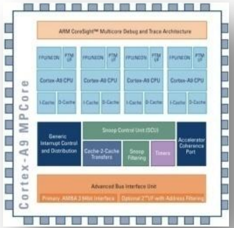


Why Adaptivity?

- Increasing complexity of embedded systems
  - Higher requirements on autonomous behaviour
- Increasing uncertainty in use cases and resource requirements
  - Designs based on worst-case prior information unfeasible



- Hardware development makes adaptivity a possibility
  - Reconfigurable hardware
  - Power saving technologies



- Hardware development increases the need for adaptivity
  - Multi- & many-core platforms
  - Variability of 10-20 nm chips
- Hardware development makes adaptivity more complicated
  - High performance on, e.g., multi-cores, for communication-heavy applications requires careful optimization and complicates on-line modifications

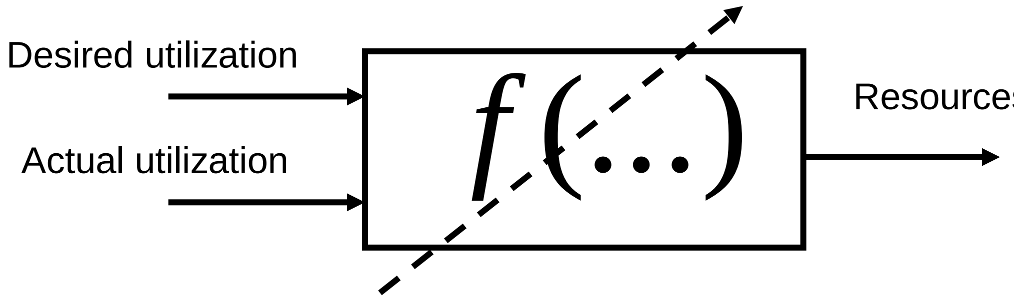
Definitions

"An embedded system is ***adaptive*** *if it is able to adjust its internal strategies to meet its objectives"*

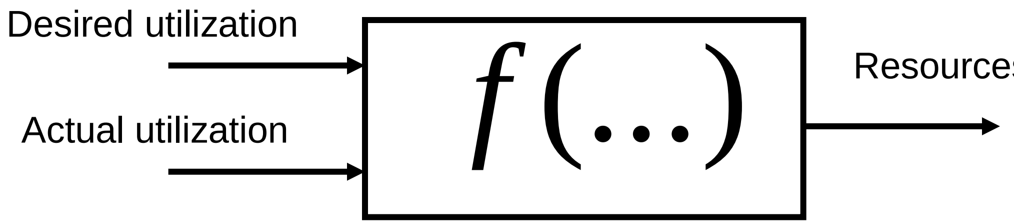
"An embedded system is ***robust*** *if it meet its objectives under changing conditions without modifying its internal strategies"*

Adaptivity - Confusion

- Adaptivity in the control community



- Adaptivity in the embedded system community



Motivation for Adaptivity

- Cope with uncertain resource requirements (CPUs, network, memory, …)
  - Unknown resource requirements
  - Varying resource requirements
  - Changes in total workload (multiple applications)
- Cope with uncertainties in resource availability
  - Changes in the amount of resources (# cores, # nodes, clock frequency, …)
    - To save power, minimize heat, ….
  - Changes in the quality of resources (network variability, ….)

Goals for Adaptivity

- Maximize the service delivered with a fixed level of resources
- Minimize the resources used while maintaining an acceptable service level
- Increase dependability
  - Reliability, safety, availability, maintainability, ….

Problems of Adaptivity

Adaptivity can introduce new problems:

- The adaptation mechanism itself consumes resources
- Harder to provide formal guarantees about the system
- Adds to the complexity
- May complicate the design process (modeling, V&V, …)
- Sensors and actuators are necessary
- Models are necessary
  - Of the system that we adapt
  - Of the adaptation mechanism itself
- Types of models not evident

Outline

- Adaptivity in Embedded Systems
- **Resource Management for Multimedia Dataflow Applications on Multi-core Platforms**

ACTORS

- Adaptivity and Control of Resources in Embedded Systems
- EU FP7 STREP project
  - 2008-2010
  - Coordinated by Ericsson (Johan Eker)
  - Lund University, TU Kaiserslautern, Scuola Superiore Sant’Anna di Pisa, EPFL , AKAtech , Evidence

Example: Cellular Phones Today

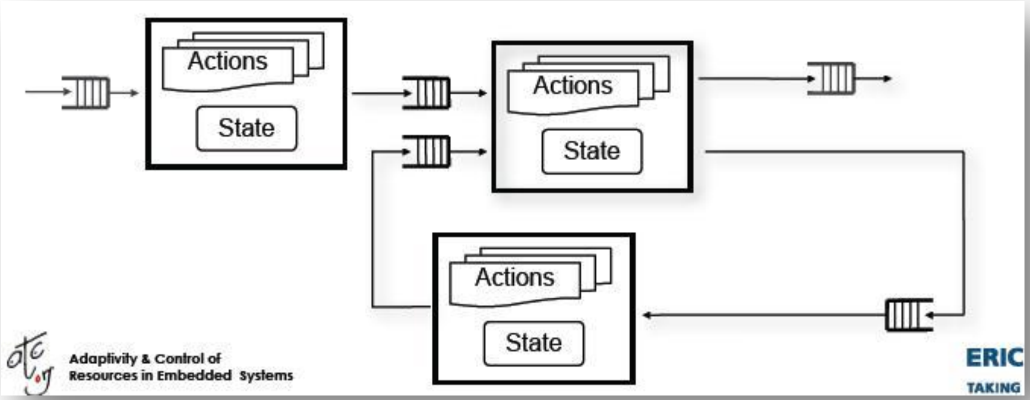
- Code Size
  - 15-20 Millions line of code
- 3-4 h build time
- Compiled into *one* program that runs from flash
- Around 100 threads with varying real-time criticality
- No static analysis
- Over-provisioning of resources to cater for worst-case not an option
- Many hundreds of parallel developers
- Certain time-critical parts hand-coded in machine language

Example: Cellular Phones Tomorrow

- Multimedia streaming and processing increasingly important
  - Multiple simultaneous streams
- Large dynamic variations in use cases and QoS demands
  - Dynamic adaptation necessary
  - Performance and power consumption reasons
- More advanced processors, e.g. ARM11
  - Multicore for performance and power
  - Powerful and complex instruction sets
  - Generation of efficient code an even higher challenge than today
- Heterogeneous
  - Open OS (Android, Linux, …)
  - Hetreogeneous hardware (ASICs, multicore, hardware accelerators)

ACTORS: Key Ingredients

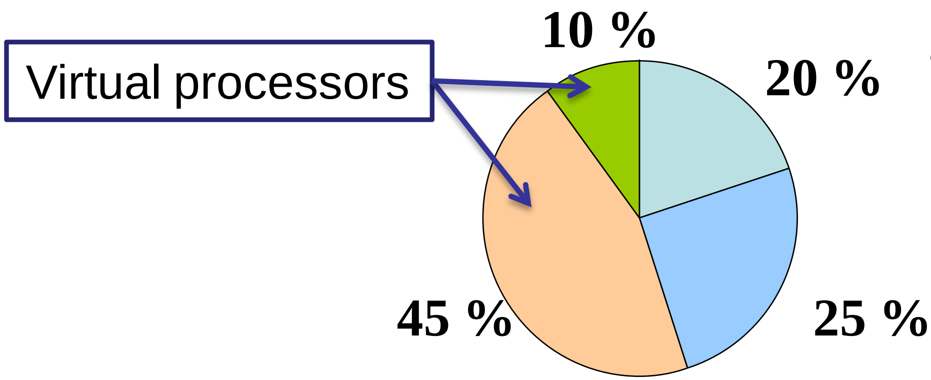
1. Data-Flow Programming



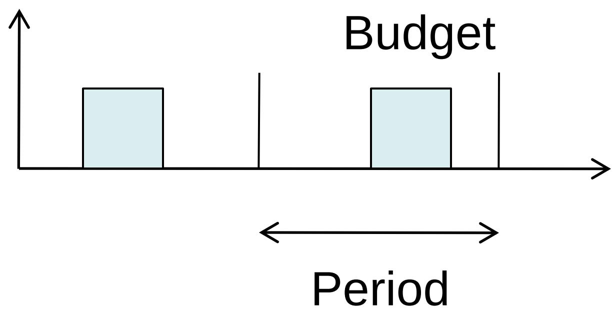
2. Feedback-Based Resource Management
   – Control how much CPU resources that are allocated to different applications based on feedback from resource utilization and achieved QoS

ACTORS: Key Ingredients

3. Reservation-Based Scheduling



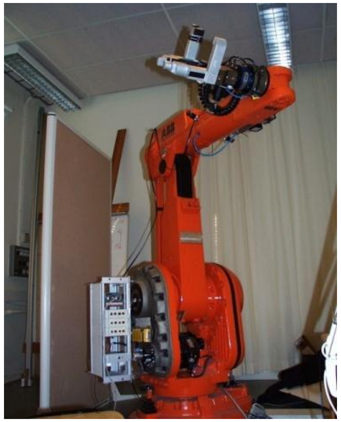
- Periodic Bandwidth Servers
  - Constant Bandwidth Server



4. Multicore Linux Platforms
   - ARM 11, x86

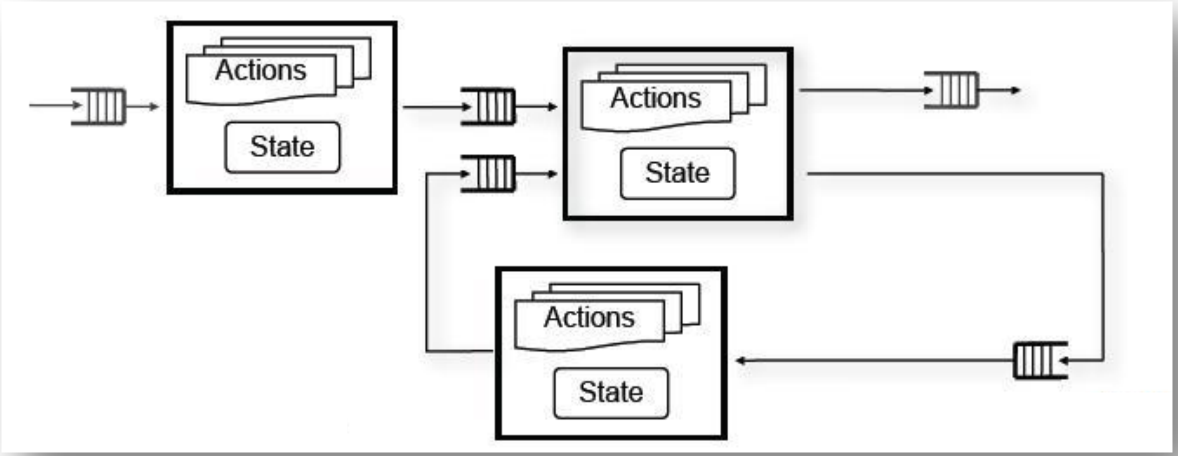
Three Demonstrators

- Media Streaming in Mobile Terminals
  - Conversational video
- Feedback Control
- High-Performance Video

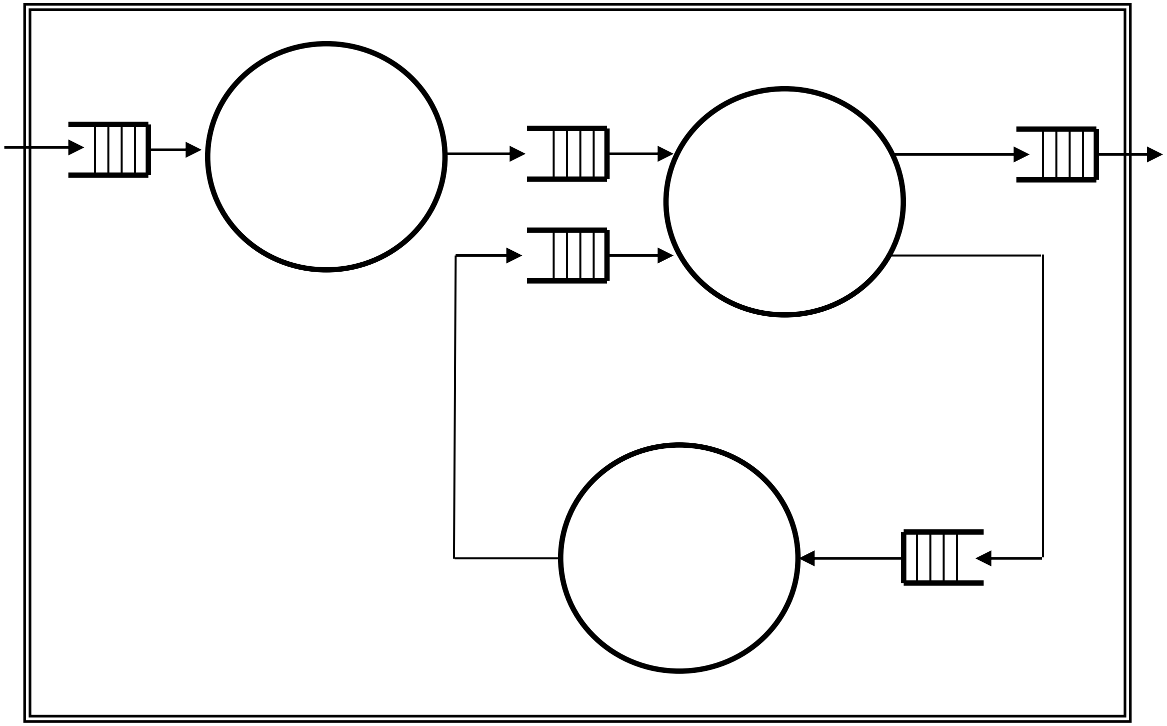


ACTORS: Dataflow Modeling

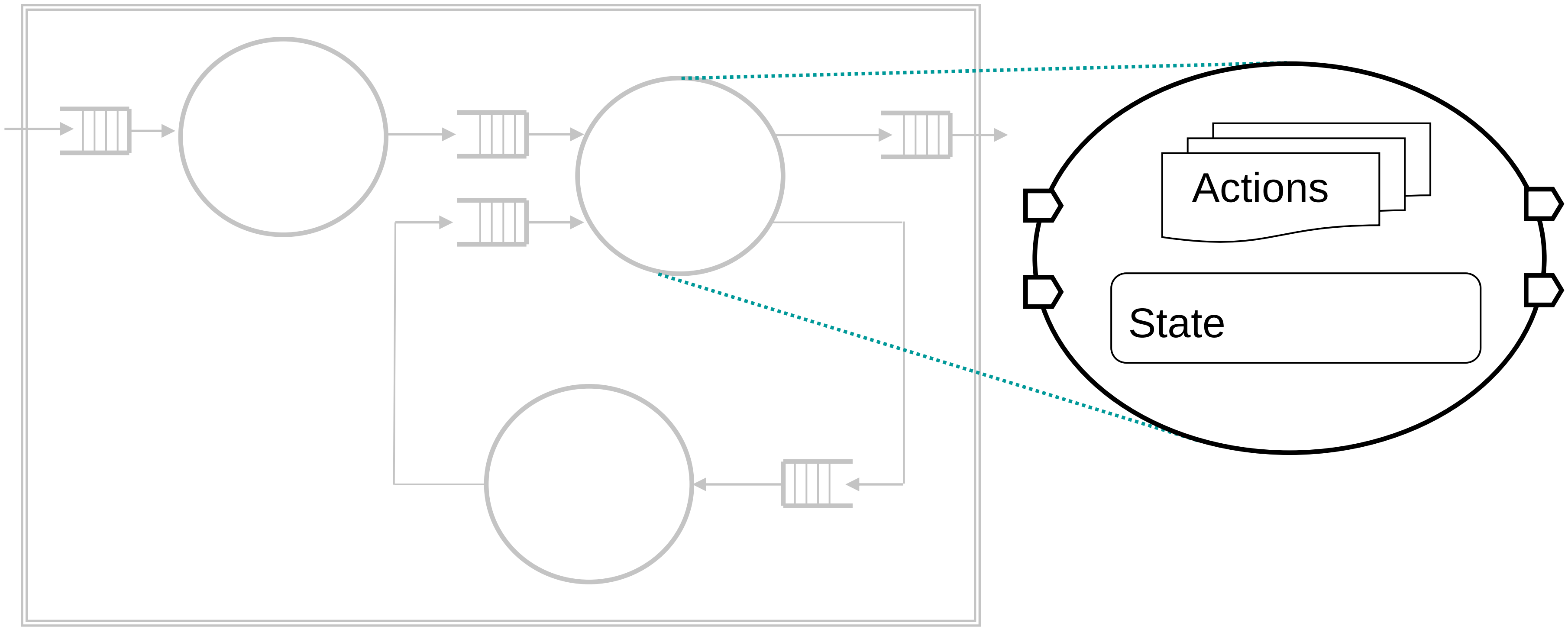
- Data flow programming with actors (Hewitt, Kahn, etc)
  - Associate resources with streams
  - Clean cut between execution specifics and algorithm design
  - Strict semantics with explicit parallelism provides foundation for analysis and model transformation
- CAL Actor Language (UC Berkeley, Xilinx) http://opendf.org
  - Part of MPEG/RVC



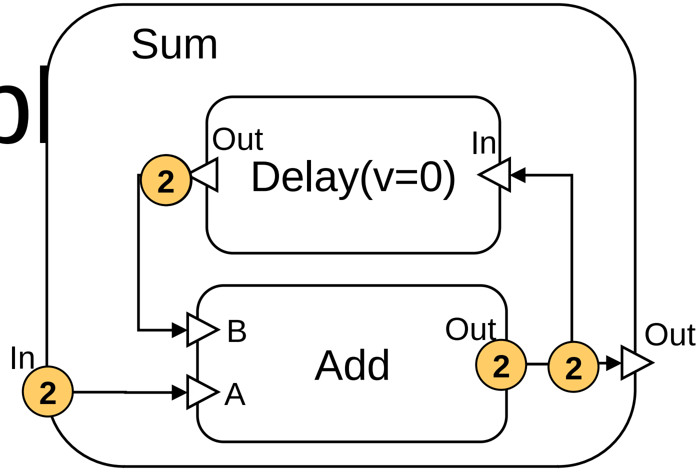
Actor Network



Actors & Actions



A small exampl

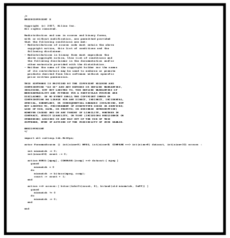



```
actor Add() A, B ==> Out:
  action A:[a], B:[b] ==> Out:[a + b] end
end
```

```
actor Delay(v) In ==> Out:

  A1: action ==> Out:[v] end
  A2: action In:[x] ==> Out:[x] end

  schedule fsm s0:
    s0 (A1) --> s1;
    s1 (A2) --> s1;
  end
end
```

```
network Sum() In ==> Out:

entities
  add = Add();
  delay = Delay(v=0);

structure
  In --> add.A;
  delay.Out --> add.B;

  add.Out --> delay.In;

  add.Out -- > Out;
end
```

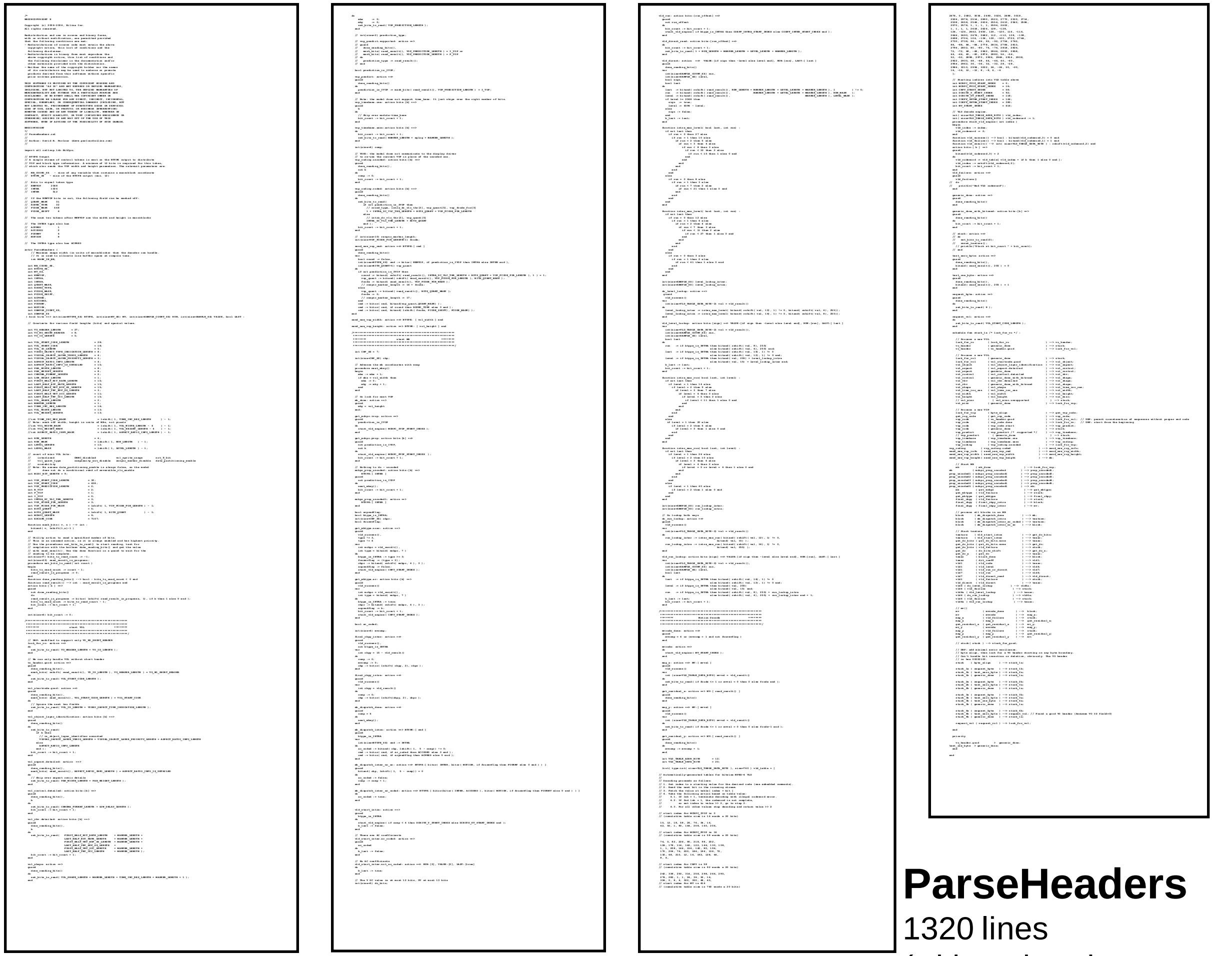
Real-life examples



Compare

23 lines

(without header comments)



ParseHeaders

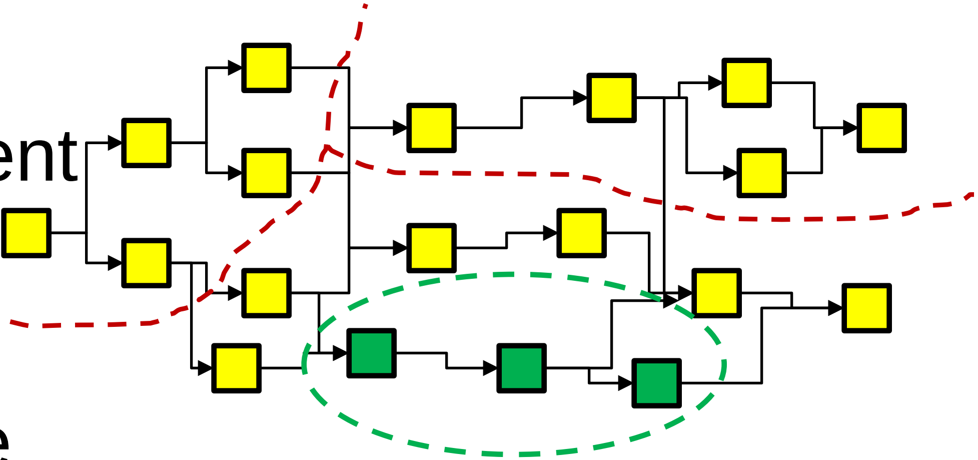
1320 lines

(without header comments)

`http://opendf.svn.sourceforge.net/viewvc/opendf/trunk/models/MPEG4_SP_Decoder/`

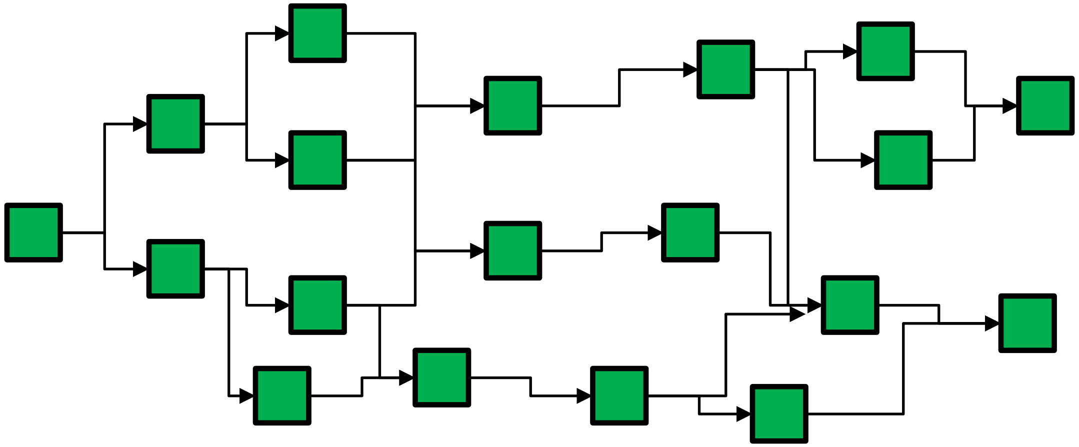
Dynamic CAL Programs

- Most actors have a data- or time-dependent behavior
- Static analysis and scheduling impossible
  - Run-time best-effort scheduling
- Some actors have a static behavior
  - The corresponding sub-network can analyzed and scheduled



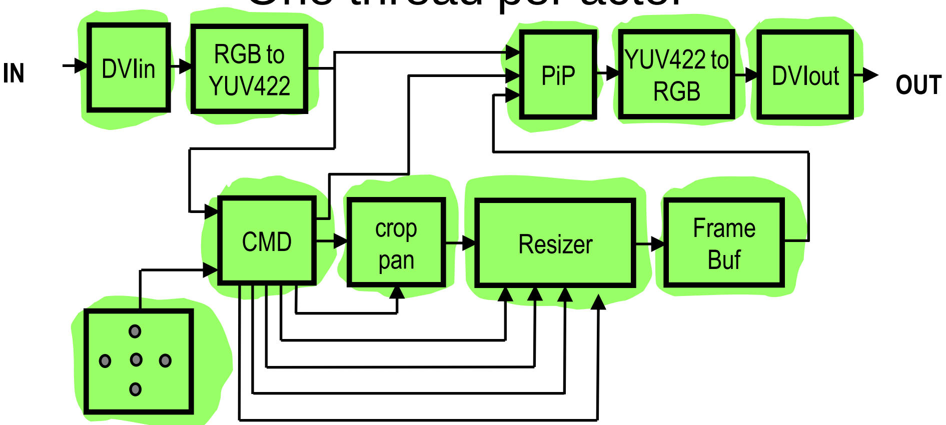
Static CAL Programs

- All actors have a static behavior
- Schedulability analysis of the entire network possible
- For example, hard-real time multi-core scheduling that guarantees end-to-end deadlines

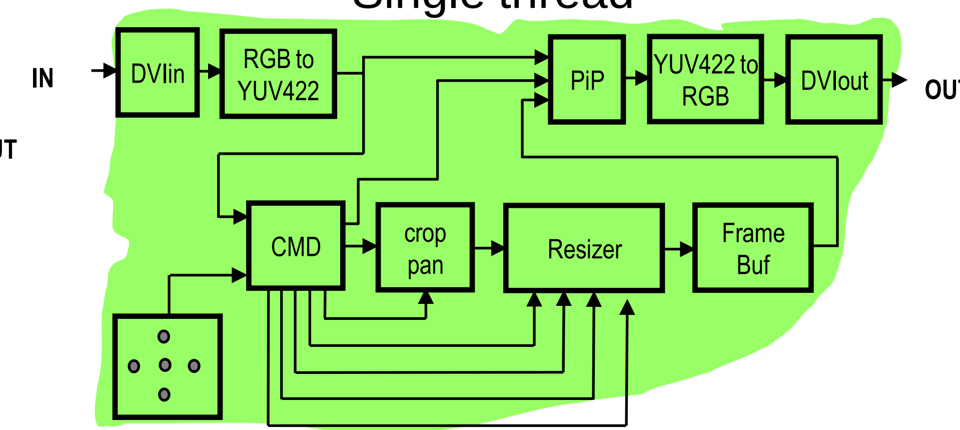


Execution of Actor Networks

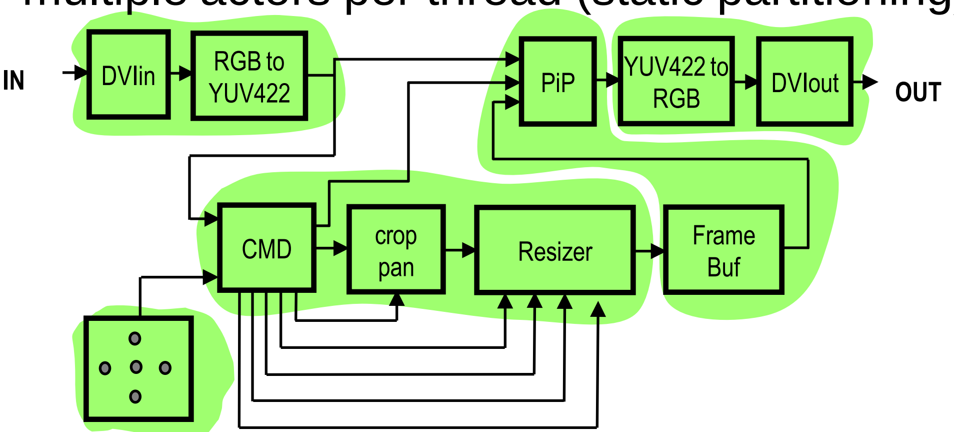
One thread per actor



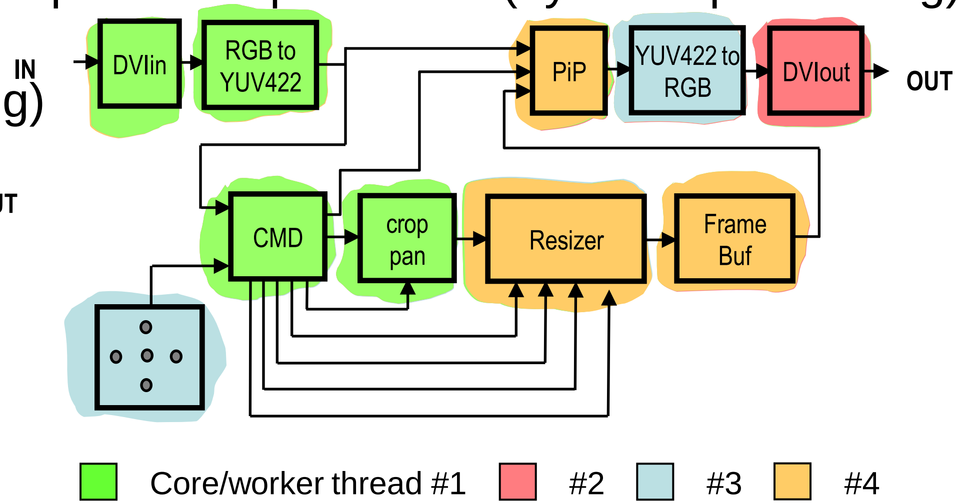
Single thread



multiple actors per thread (static partitioning)

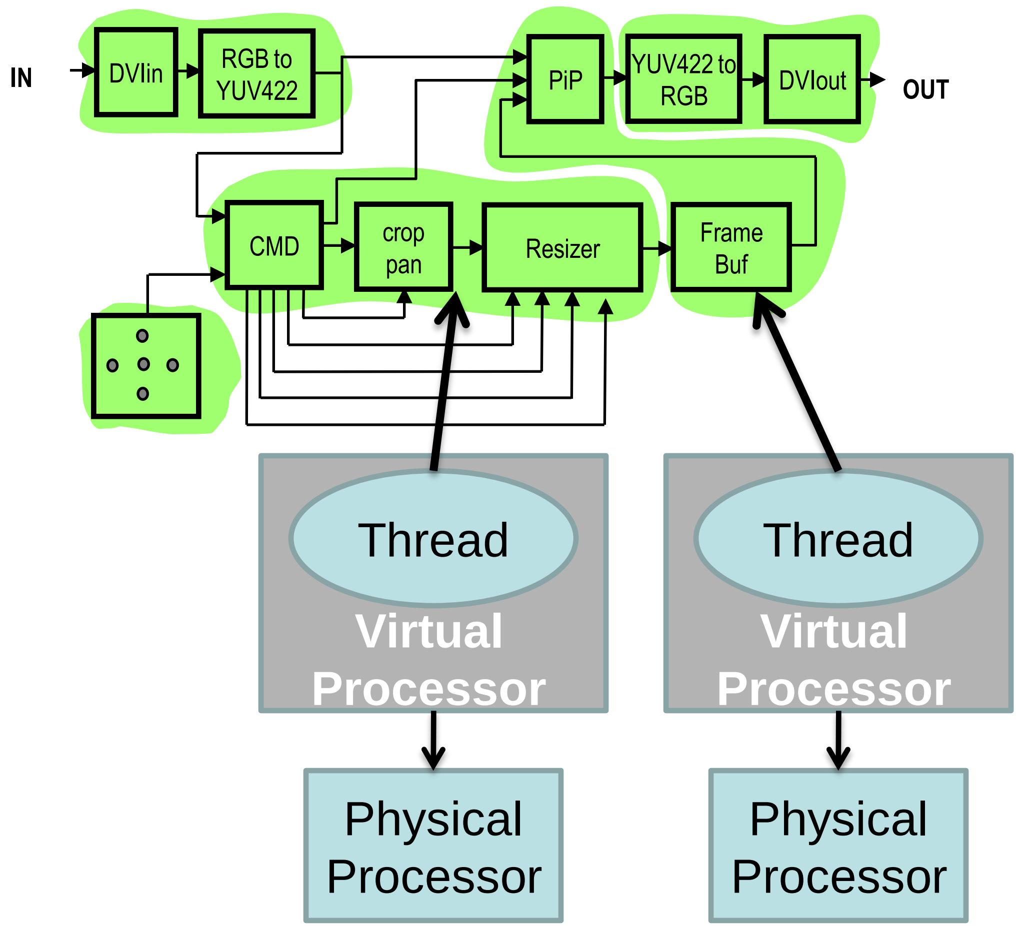


multiple actors per thread (dynamic partitioning)



CAL Run-Time System

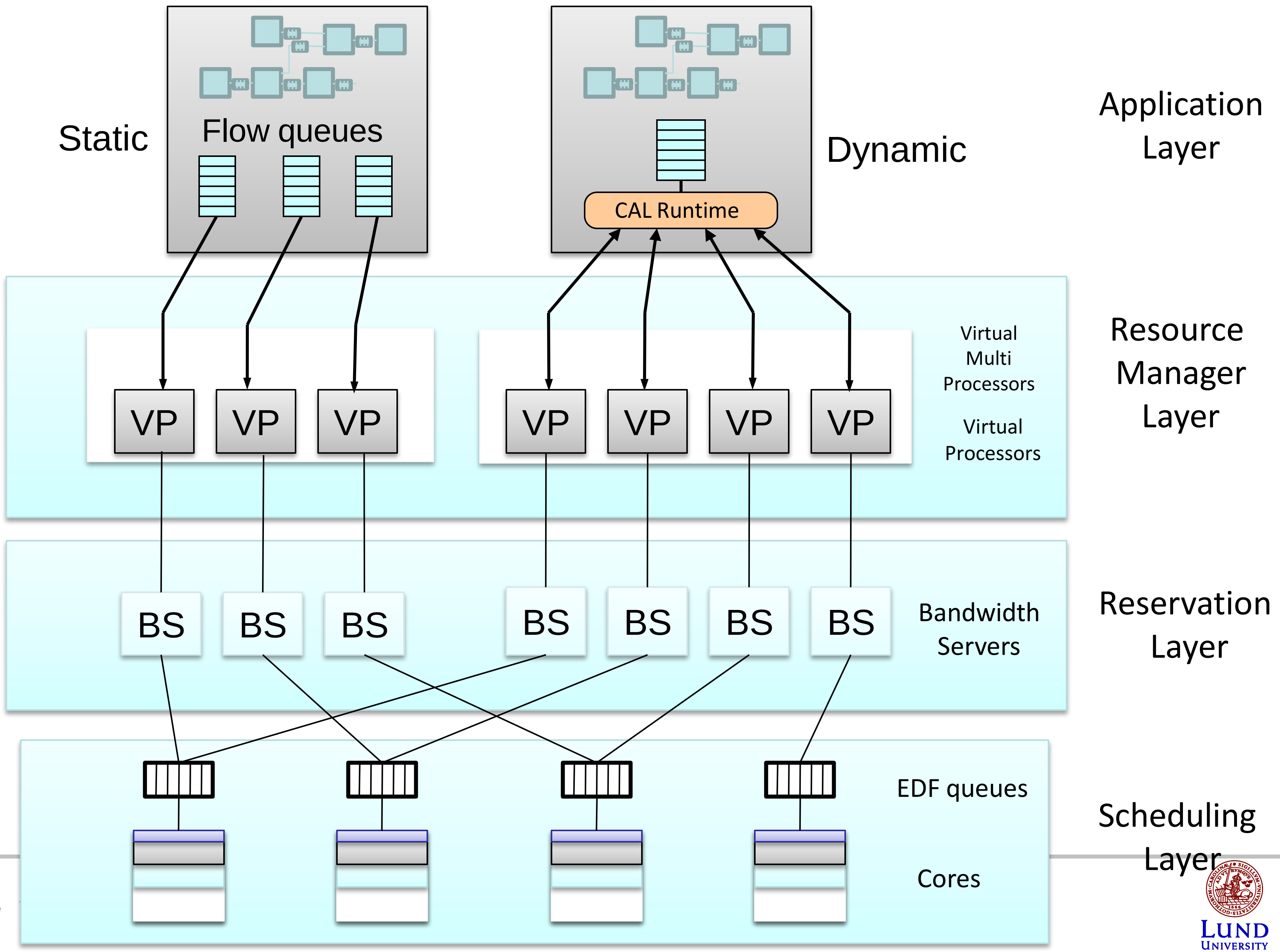
Static partitioning based on off-line analysis

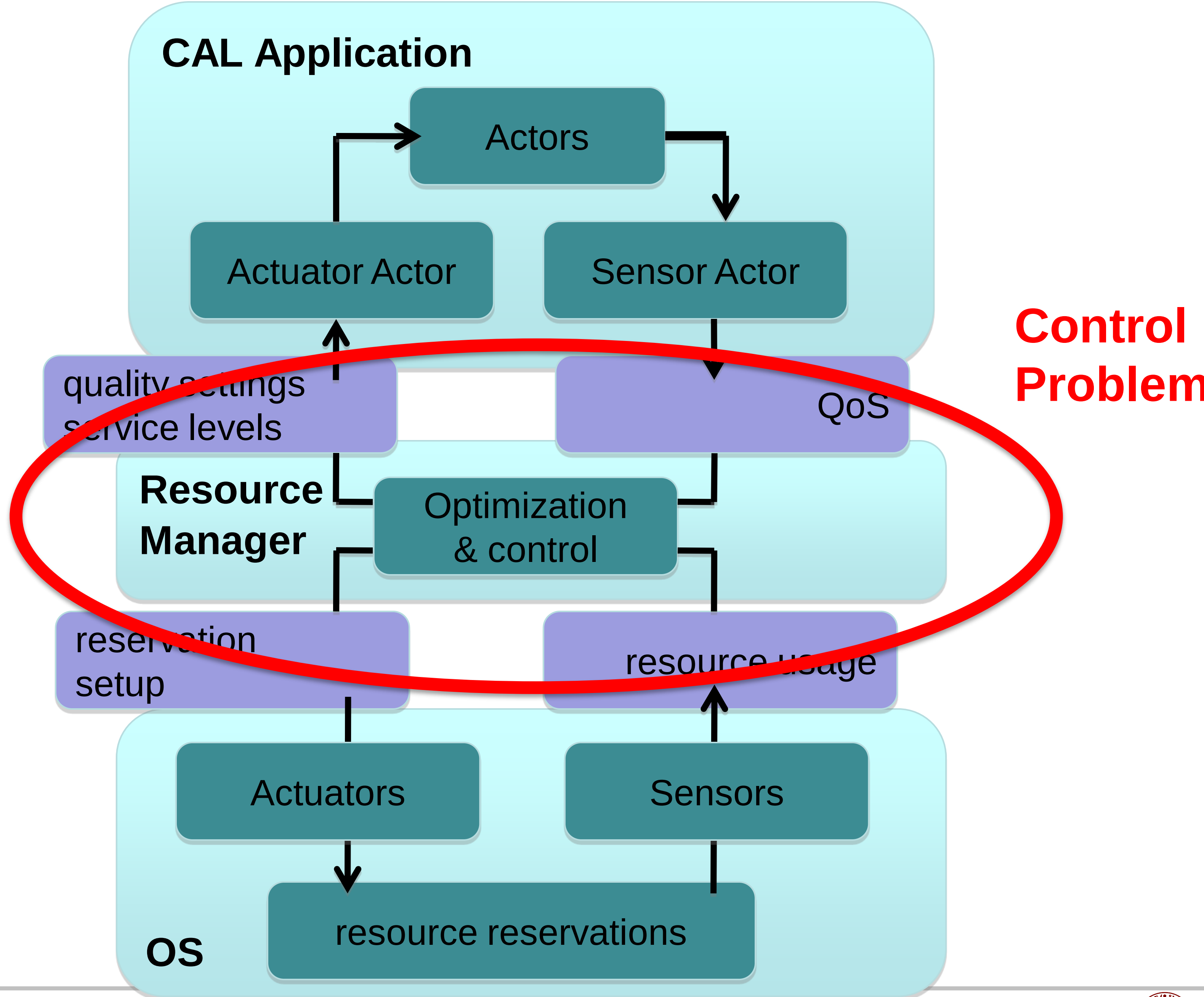


Virtual Processors

- Hard Constant Bandwidth Servers (CBS)
- Partitioned virtual processors (reservations) → a multi-core application cannot execute on a single virtual processor
- SCHED_EDF new scheduling class for Linux
  - – Evidence

Structure


Static
Flow queues
Dynamic
CAL Runtime
Application Layer
VP
Virtual Multi Processors
Virtual Processors
Resource Manager Layer
BS
Bandwidth Servers
Reservation Layer
EDF queues
Cores
Scheduling Layer


CAL Application
Actors
Actuator Actor
Sensor Actor
quality settings
service levels
QoS
Resource
Manager
Optimization
& control
reservation
setup
resource usage
Actuators
Sensors
resource reservations
OS
Control
Problem

Resource Manager Purpose

- Manage resource for all ACTORS applications
  - Set up reservations
  - Use feedback to adjust reservation sizes
  - Select appropriate service levels of the applications

- *Applications need to be adaptive*
  - Should provide multiple service levels with different quality and resource demands
  - Publish quality levels to the resource manager

- Additionally importance values for applications
  - Set by user or system integrator

Available Information - Static

- From the applications:
  - Service Level Table (~ *model*)

Service Level	QoS	BW Requirement	BW distribution	Timing Granularity
0	100	250	25-25-25-25	20 ms
1	75	180	25-25-25-25	20 ms
2	40	120	25-25-25-25	20 ms

- BW Requirement - Total BW required by the application
- BW distribution - Initial distribution of the BW among the cores
- Timing Granularity

Available Information - Static

- From the system administrator:
  - Importance values
    - The relative importance among the applications in the case of an overloaded system

Appl.	Importance
Appl 1	10
Appl 2	20
Appl 3	100

Available Information - Dynamic

- From applications: ☺ ☹
  - – Happiness value
    - Binary value indicating whether the QoS obtained by the application corresponds what can be expected for the current service level

Available Information - Dynamic

- From OS/Reservations system
  - Used budget
    - Average used budget over a specified measurement period per virtual processor
  - Hard reservations
    - Number of times that the used budget exceeds the assigned budget over a specified measurement period per virtual processor
      - Budget exhaustion = application throttling

Resource Manager Tasks

- Service Level Assignment
  - Select the service levels of the applications taking into account the application importance and the total amount of resources available (total CPU bandwidth)
  - When applications arrive or terminate, when the amount of available resources changes, ….
- Mapping
  - Map the virtual processors to physical processors (cores)
- Bandwidth Distribution
  - Distribute the total bandwidth of each application onto the virtual processors of the application according to some criteria
- Bandwidth Adaptation
  - Adjust the allocated bandwidth for each virtual processor based on measurement of the used budget, hard reservations and happiness
  - May lead to an update of the service level tables and trigger a new service level assignment

Feedback Control Options

- Within the RM
  - Feedback only
    - Challenge - how to respect application importance and limited resource availability
  - Feedforward + feedback
    - Feedforward = service level assignment and bandwidth distribution solved using optimization
    - Feedback = bandwidth adaptation
    - Challenge - how to handle uncertainties and variations
- Within applications
  - Adjust the amount of resources used to meet a requested service/quality level

RM in Android

- Ericsson has deployed an early version of the ACTORS RM in the Android OS
- The RM allows applications to register their
  - service levels
  - resource requirements
  - threads
- The RM changes the application's CPU share based on
  - measured CPU usage
  - application importance

TrueTime Resource Manager

- Matlab/Simulink toolbox for simulation of real-time kernels and networks developed at Lund Unniversity since 1999
- Extended within ACTORS to support
  - Partitioned multi-core scheduling
  - Hard CBS servers + EDF scheduling
- Flexible, yet realistic, platform for experimenting with the decision logic of resource management

Conclusions

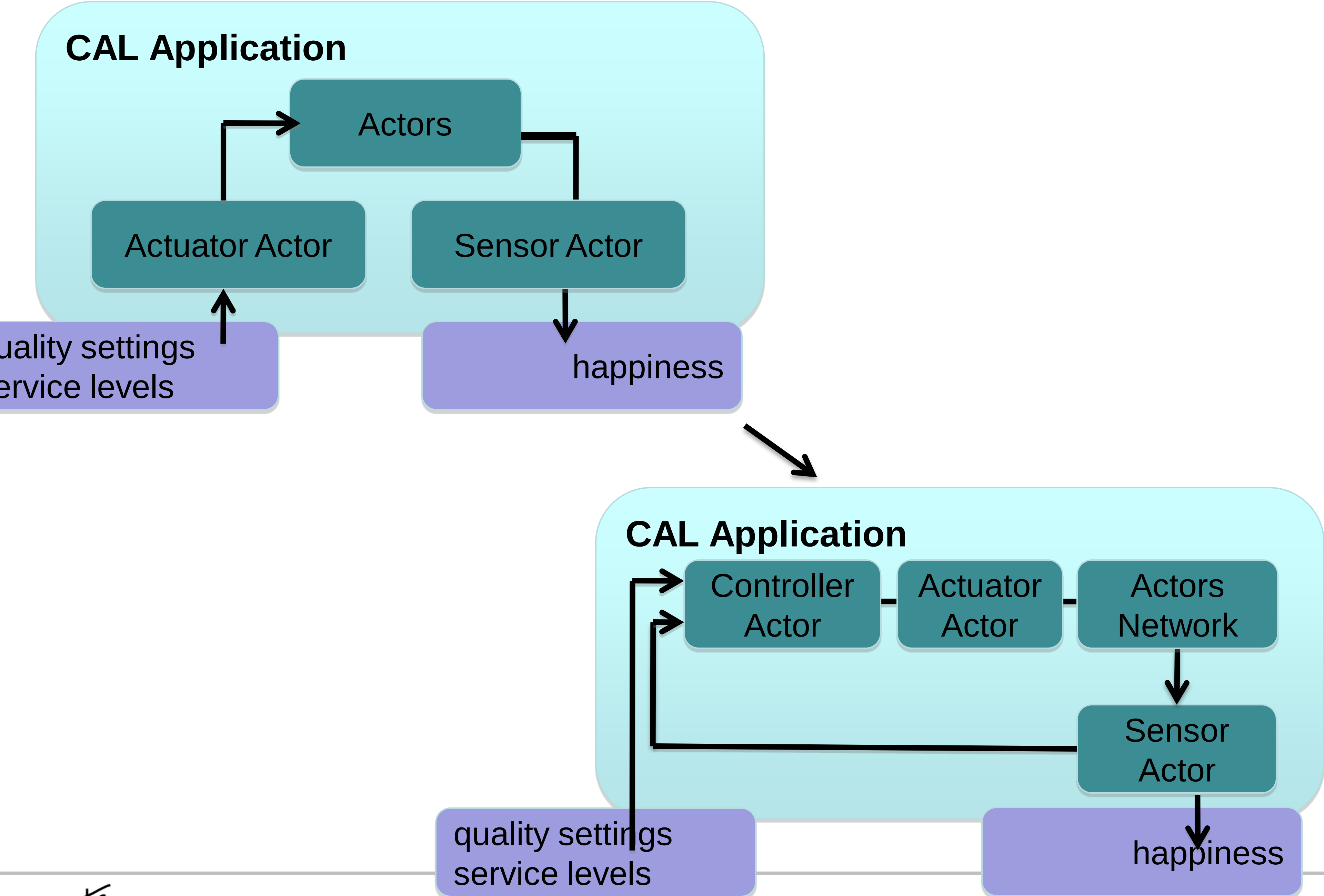
- Feedback-based resource management (scheduling) for embedded computing systems a challenging area for control
- Still in its infancy
- Huge potential (cp. process control)
- Hardware development increases the need for more dynamic approaches than what is current practice
- Interesting cross-disciplinary area

More Information

- http://www.actors-project.eu/

Conclusions

- CPU time reservations with adaptation for multi-core dataflow applications
- Several challenges:
  - High performance on multi-cores for communication-intensive applications requires careful optimization and complicates on-line adaptations
  - Not entirely obvious how the resource manager should operate
  - What information can be expected from the applications?


CAL Application
Actors
Actuator Actor
Sensor Actor
uality settings
ervice levels
happiness
CAL Application
Controller Actor
Actuator Actor
Actors Network
Sensor Actor
quality settings
service levels
happiness